1h

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/924,272	08/07/2001	Paul Metzgen	174/193	4896

36981 7590 11/01/2005

FISH & NEAVE IP GROUP
ROPES & GRAY LLP
1251 AVENUE OF THE AMERICAS FL C3
NEW YORK, NY 10020-1105

EXAMINER
VU, TUAN A

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 11/01/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/924,272	METZGEN, PAUL
	Examiner	Art Unit
	Tuan A. Vu	2193

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 October 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 10/3/2005.

As indicated in Applicant's response, claims 1, 9, 20, 24, 28, and 34 have been amended. Claims 1-37 are pending in the office action.

Claim Objections

2. Claim 1 is objected to because of the following informalities: the use of the verb 'configures' (re claim 1, line 9) does not fit the proper and commonly accepted usage thereof. A piece of data is usually not a subject performing the act of directly configuring of another object but should only be a means for some intelligence like a program, a machine or a developer to configure another object. This misuse of 'configures' will be treated as though the recited 'configuration data' is directly responsible of a configuration of another resource.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 9-19, 20-23, 28-33 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Specifically, claim 9 recites 'hardware execution run-time'. Scanning the content of the specifications, it is found that there are instances reciting the word 'execution' (e.g. all of which are related to loop, parallel execution of control flow blocks or software constructs like a shared lock, a logic condition) but hardly anywhere in the specifications is such term associated explicitly – and with reasonable clarity -- with any form of hardware, and even less with both 'hardware' <u>and</u> a 'run-time' thereof. The title of the invention and the summary thereof is about converting format from one form to another like in a compiler; and in the specifications, it is very hard to be conveyed when such compiler activity would end and when --or what, a representation of what is called a finite executable would start running at the end of a assumingly completed compilation process; hence the limitation as to an *execution runtime* would be without actual basis that would necessarily direct it to be occurring outside of a compiler scope as titled. The claim is hence not enabling one skill in the art to construe a claimed subject matter owing to the lack of description thereof; and this deficiency leads to the possible conclusion that the inventor has no possession of the claimed invention. The above limitation would be treated as mere execution of software constructs during a run time in conjunction with implementing some target being possibly hardware or software resource.

Claims 10-19 are also rejected for not remedying to the deficiency of the base claim.

Claims 20 and 28 also recite 'hardware execution runtime', and are also rejected for the same reasons. This limitation will be treated as execution of software constructs during a run time in conjunction with implementing some target being possibly hardware or software resource.

Claims 21-23, and 29-33 are also rejected for not remedying to the deficiency of the base claims.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 27, 28-33, 34-36, and 37 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

> The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

Claim 27 recites a method for mapping software constructs into hardware constructs comprising parsing and mapping variable into a set of wires being a hardware construct. As interpreted, the fact of parsing can be a mental process and the mapping can be a mental task done via use a pen and paper apparatus. There is no clear teaching that the actions so claimed require a tangible hardware support. Absent any tangible embodiment, the claim therefore amounts to an abstract idea for failing to provide a concrete and tangible result; and is rejected for leading to a non-statutory subject matter as set forth above in the Practical Application Test requirement.

Claim 28 recites a programmable logic *resource* configured to generate a control flow and make hardware execution run-time decisions. As construed from reading the specifications, every single execution described therein proves to be execution of a software programming

construct or code; and nowhere is there any hardware execution per se; and this has been set forth in the USC 112 rejection from above. Further, the instances reciting the term *resource* as repeatedly found in the specifications do not make it clear that this resource is a tangible hardware resource. Indeed, the so-called *resource* as found in the disclosure represents or relates one of the following: a lock or a shared software construct, an output variable, a logic array, a speculative condition, a left-most or loop entity, or memory value.

Since the claim recites a resource perceived as software entity without further providing a hardware embodiment so as to support the execution of such entity, the claim fails to fulfill the requirements set forth by the above Practical Application Test from above. Absent any tangible embodiment, the claim therefore amounts to an abstract idea for failing to provide a concrete and tangible result; and is rejected for leading to a non-statutory subject matter.

Claims 29-33 are rejected for not remedying to the base claim.

Claim 34 also recites a programmable logic resource comprising constructs mapped from software constructs; and lack description of any tangible elements to support such resource. Based on the rationale set forth above in regard to the term *resource* being described as no more than a software resource throughout the disclosure, this claim is also rejected for leading to a non-statutory subject matter. Claims 35-36 are rejected for not remedying to the base claim.

Claim 37 recites a hardware construct comprising a set of wires being a mapping of some software variable. Thus, the so-recited *construct* amounts to no more than a set of variables; hence the claim is not providing hardware support to or to embody the elements recited as constructs or variables. Absent any tangible hardware support, the claim fails to yield a tangible

result and amounts to a mere non-applicable abstract idea; and is rejected for leading to a non-statutory subject matter.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.
>
> Note: 35 U.S.C. § 102(e), as revised by the AIPA and H.R. 2215, applies to all qualifying references, except when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. For such patents, the prior art date is determined under 35 U.S.C. § 102(e) as it existed prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. § 102(e)). The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that fonn the language.

8. Claims 1, 2, 4-5, 7-8, 24-26, and 34-36 are rejected under 35 U.S.C. 102(e) as being anticipated by Panchul et a1., USPN: 6226,776 (hereinafter Panchul).

As per claim 1, Panchul discloses a method for generating hardware configuration data from software constructs, the method comprising: parsing high-level software programming code, wherein the code is transparent with regard to the hardware resources and hardware configuration (e.g. Fig. 2; Fig. 3-22 - Note: compiling a C program instructions to provide hardware definition language and register transfer form code is equivalent to parsing and that the C-instructions are abstract data structures remote to the layer of hardware resources, hence transparent to hardware resources and configuration); and compiling hardware configuration data directly from the high-level software programming code (e.g. Figs. 3-22; *ANSI C ... compiled*

into - col. 13, lines 32-37); wherein the hardware configuration data configures a programmable logic resource (e.g. *reg, wire, pa0-pa16, RAM32x1* -- Fig. 7B1-B4 --Note: Verilog driven software constructs to implement hardware constructs like RTL reads on hardware configuration data configuring programmable logic resource) .

As per claim 2, Panchul discloses hardware configuration data implementing blocks (e.g. HDL - col. 13, line 1 to col. 14, line 65 - Note: the organizing of hardware circuitry via HDL or Verilog specification implicitly discloses organizing the hardware configuration data in blocks - see Panchul, col. 8, lines 36-40, *always block* - Fig. 8B).

As per claim 4, Panchul discloses pipelining (e.g. Fig. 8B)

As per claim 5, Panchul discloses a FPGA (e.g. col. 12, lines 42-49; col. 13, lines 53-63).

As per claim 7, Panchul disclose runtime decisions for parallel execution (e. g. Fig. 3A-B; Fig. 14A-B; *HDL functional blocks ... independently of each other* - col. 21, line 28-46 - Note: compiling into hardware configuration, determine which HDL blocks for being synchronously executing reads on hardware blocks being generated and making independent runtime decisions while parallel executing).

As per claim 8, Panchul discloses C code (e.g. Fig. 2, 3A, 4A, 5A, 6A).

As per claim 24, Panchul discloses a method for mapping software constructs directly into hardware constructs, the method comprising

mapping software constructs directly into a block of logic operations using a software/hardware compiler (e.g. Fig. 2; *ANSI C ... compiled into* - col. 13, lines 32-37) wherein the logic operations configure a programmable logic resource (e.g. *reg, wire, pa0-pa16, RAM32x1* -- Fig. 15B1-B7 --Note: Verilog/HDL derived or driven program constructs - or

blocks of logic operations implementing a logic of some hardware circuitry functionality to implement hardware constructs - reads on hardware configuration data configuring programmable logic resource);

coupling input to the block and coupling output to that block (e.g. *Ram32xl6, Ram32xl, DivEx* - Fig. 5B1; Fig. 7B3).

As per claims 25 and 26, Panchul discloses input and output environment including wires implementing variables, pointer, expressions, and *reset* and *done* signal (e.g. Fig. 3-22; *if(reset)* - Fig. 7B3; *state=* ... , *end-* Fig. 7B3) and control flow (*controlflow* - col. 7, lines 22-43;Fig. 5A-B).

As per claims 34-36, these are programmable logic resource respective versions of claims 24-26, the programmable limitation being addressed by Panchul's programmable logic (col. 12, lines 42-49; col. 13, lines 53-63); and are rejected with the corresponding rejection in claims 24-26 as set forth therein.

9. Claim 20 is rejected under 35 U.S.C. 102(e) as being anticipated by Killian et al., USPN: 6,477,683.

As per claim 20, Killian discloses optimizing hardware generated by software-to-hardware compiler, comprising:

locating during compilation at least one expression wherein said at least one expression is used more than once, and using a single set of hardware resources to implement such expression (e.g. *pattern, replaced ... single instruction, how often, hardware estimator, set of TIE instructions* - col. 19, line 47 to col. 20, line 9; col. 27, lines 10-16- Note: detecting pattern in tree so that a set of instructions from hardware estimator can replace to improve hardware

efficiency reads on using a single set of hardware resources to implement the expression if expression is used more than once);

and during hardware execution runtime (col. 27, lines 10-16 – Note: TIE generating state and event while running generating dynamic flow graph reads on runtime regarding execution of a state machine analysis represented in the graph),

selecting at runtime instances that will have access to the single set of hardware resources (e.g. col. 19, line 47 to col. 20, line 9; col. 27, lines 10-16 - Note: the partitioning of hardware in blocks by a HDL mapping and optimizing replacement reads on analysis of a TIE runtime eventuality wherein executing instruction instances that will access a set of hardware resources as taught above).

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 3, 27, and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Panchul et a1., USPN: 6,226,776.

As per claim 3, Panchul does not disclose data wires and a computed wire to denote the variable is valid. Panchul discloses wires being mapped to variables and associated pointers to represent a variable, expression or a function (e.g. col. 5, line 4-45; Fig. 1 lA-B) and a set of wires to enable a variable (LCD) with wire specialized to represent data (e.g. Fig. 20B2, *input ... A or D or WE, wire [4:0]* - Fig. 5B1), hence has suggested the creating of a wire or pointer

structure representing a declared variable or function and wires for data and input control bit. Further, as illustrated by Panchul, it was a well-known concept that a set of data being passed through a memory port or a circuit gate and such passage is being enabled by a valid or enable Bit was a known concept in the m4 of hardware design; and this can be shown as example in Panchul's pin WE (gate *FDCE* - Fig. 23C-2, Fig. 25C-1). Hence, since a set of wires or pointers are used for representing data or variables as suggested by Panchul's HDL and its parsing scheme, it would have been obvious for a skill in the art to represent circuitry wires according to HDL specification as suggested by Panchul above so that some bit in the set of data (or one wire among the set of wires) would be representing the existence or validity of the variable or a non-null pointer, and that some other bit/wires represent the data for that variable according to the well-known concept above. The motivation would be because, wires are purported for transporting data through a gate device and this requires control and according to the above well-known concepts, such control, using one bit or wire to enable or indicate validity/permissibility, would allow such data lines or wires to be checked by the device or gate before data can be allowed to go beyond such gate, or to be put for storage, or read/written via memory port; and this is according the well-known concept of the enabling pin as illustrated by Panchul's above example.

As per claim 27, Panchul discloses a method of mapping software constructs into hardware constructs, comprising parsing the software constructs (e.g. col. 23, lines 63-65); mapping a software construct variable into a hardware construct comprising a set of wires (e.g. col. 5, line 4-45; Fig. 11A-B, Fig. 20B2; *input ... A or D or WE, wire [4:0]* - Fig. 5B1). The limitation about the wires representing a variable definition/computation state and the value of

the variable in this claim corresponds to that of claim 3, hence is rejected using the same rationale as set forth therein.

As per claim 37, this is the hardware and programmable logic version of claim 27, hence is rejected using the same rationale as set forth therein; the programmable limitation being addressed by Panchul's programmable logic (col. 12, lines 42-494 col. 13, lines 53-63).

12. Claims 6, 9-12, 16-19, and 28-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Panchul et al., 6,226,776, in view of Killian et al., USPN: 6,477,683.

As per claim 6, Panchul only discloses optimizing and pipelining of loops (e.g. col. 4, lines 51-63; col. 27, Table, lines 45-55; Fig. 8B); but does not teach hardware capable of speculative execution. The method of applying high-language programming optimization when implementing compiler analysis to support pipelining and loop scheduling as endeavored by Panchul (e.g. col. 4, lines 61-67; Figs. 19, *optimized* – table, lines 45-55) was further enhanced by speculative execution of Killian. Killian, in a method to compile C-code specifications and convert HDL language into target hardware implementation analogous to Panchul, discloses pipelining as well as speculative by compiler directives and shared memory (e.g. *speculation* - col. 13, line 44 to col. 14, line 4). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the optimization techniques by Panchul the compiler-driven speculation as by Killian because according to Killian, these speculative executions can avert data fetching exceptions which would otherwise require unwanted correction handling resources (e.g. col. 13, line 44 to col. 14, line 4).

As per claim 9, Panchul discloses a method for generating hardware exploiting parallelism by making decisions at hardware execution runtime (Note: generating state and event

while running and generating dynamic flow graph based on RTL – Figs. 9 - reads on runtime regarding execution of a state machine analysis represented in the graph), the method comprising:

generating a control flow in the hardware configured in blocks (e.g. control flow, HDL, Verilog - col. 7, lines 22-43; Fig. 5A-B; col. 8, lines 36-40; *always block* - Fig. 8B);

making determination, i.e. making decisions-- to exploit parallel execution based on HDL compilation and configuration based on control flow, event or machine state - *event* and *state* read on dynamic behavior or runtime analysis -- at hardware execution runtime (Note: generating state and event while running/generating of a flow graph based on RTL – Figs. 9 - reads on runtime regarding execution of a state machine analysis represented in the graph) associated with HDL analysis (e.g. Fig. 3A-B, Fig. 12A-B, 14A-B).

But Panchul does not disclose that the control flow indicates a status for a block, status indicative for a capability for speculation. The concept as to base on an entry tag value, a control bit, a variable value, or predicate check in order to establish whether a chunk of instructions can be executed speculatively via compiler analysis was a known concept as has been evidenced from Killian's optimization using speculative execution from above. The limitation as to provide speculation to the implementation from compiling HDL into hardware target architecture set has been addressed above using Killian; hence this speculation based on a block status while analyzing control flow as mentioned above by Panchul would have been obvious in light of the known concept and of Killian's teachings for the same reasons as set forth in claim 6 above.

As per claims 10 and 11, the partitioning of HDL constructs into blocks of instructions as evidenced by Panchul (col. 8, lines 36-404 always block - Fig. 8B) was a known concept in

the HDL programming language (e.g. Verilog, VHDL, Handel-c) and in view of Panchul's teaching of determining to execute simultaneous instructions based on event or machine state analysis or to always execute a block (Fig. 3A-B; Fig. 12A-B; 14A-B; Fig. 8A) ms put forth in claim 9, the rationale to use an indication at a given point into a block of instructions in order to determine whether to execute the block via speculation has been addressed in claim 9; hence the rationale as to combine Panchul' s block with well-known concepts in association with the speculative execution as taught by Killian is herein applied to address why it would be obvious to decide to execute the block based on a tag or predicate statement evaluation.

As per claim 12, the rationale for rejection has been addressed in claim 9.

As per claim 16, Panchul teaches always blocks, hence has implicitly disclosed them to be non-mutable operations (e.g. Fig. l2B).

As per claim 17, Killian discloses replacing multiple patterns with a single instructions (col. 1 9, lines 47-65) and replacing a maximally sized match in the functions tree with one equivalent instruction (e.g. col. 27, lines 10-16). Hence, in the line as to enhancing further of Panchul's teaching of a tree traversal and pipeline techniques, it would have been obvious for one of ordinary skill in the m4 at the time the invention was made to provide on top of the variable mapping and function creation as suggested by Panchul (Fig. 3-22) the replacement techniques (or mutable instruction representing a HDL state bcing replaced) by Killian because of the same reasons as to optimize resources being recognized as well-known at the time the invention was made in the art of compiler where a excess or potential redundant resource usage is averted by a replacement policy or instruction just as taught by Killian.

As per claim 18, Panchul discloses a control flow (e.g. Fig. 5A-B).

As per claim 19, Panchul discloses implementing software in hardware (e.g. control flow, HDL, Verilog - col. 7, lines 22-43,. Fig. 5A-B,' col. 8, lines 36-404 always block - Fig. 8B).

As per claim 28, this is a programmable logic resource version of method claim 9, the programmable logic resource limitation being addressed by Panchul's programmable logic resource (col. 12, lines 42-49; col. 13, lines 53-63; Fig. 1 lA-B; Figs. 15); and is rejected with the corresponding rejection in claim 9 as set forth to address the control flow status for an operation and decision partially based on the control flow limitations.

As per claims 29-33, these claims correspond to claims 10-12, 18, and 19 respectively; hence are rejected using the corresponding rejection as set forth therein respectively.

13. Claims 13-15, and 21-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Panchul et a1., USPN: 6,226,776, in view of Killian et al., USPN: 6,477,683; and further in view of Ashar et al., USPN: 6,745,160 (hereinafter Ashar).

As per claims 13-14, Panchul does not explicitly teach deciding to execute the block speculatively and in parallel; or no data dependency between speculative and parallel execution of blocks. But Panchul discloses a tree of function/nodes (Figs. 11) and determining if an external event or a state machine is required during analysis of called function/node derived from HDL and if not then simultaneous execution of C-type functions can be effected (e.g. col. 20, line 22 to col. 21 , line 17); and non-dependency of C-type functions being simultaneously executed (e.g. col. 21, lines 28-46); hence has taught no data dependency of instructions blocks (or functions from a tree node) being paralleled.

The motivation as to use speculation in optimization of runtime execution of functions has been addressed using Killian. As for the decision to execute the block in parallel or speculatively, Ashar, in a method using netlist and VHDL to transform loop execution into a optimized and validated sequence or scheduling using speculation and parallelism as mentioned by Panchul and Killian, further discloses pipelining loops and speculative execution being determined upon analysis of control or state graph, loop invariants, and state and boundaries verification (e.g. col. 26, lines 17-29; col. 27, line 45 to col. 28, line 6). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the combination Panchul/Killian the verification ms taught by Ashar so that decision can be made to use either speculation or pipelining or loop unrolling because Ashar's techniques would provide timely validating of data being fetched for execution thus enhance optimizing success without sacrificing resources for data checking and dependencies when it is too late in the execution stage of complex loop execution (see Ashar Background).

As per claim 15, Panchul discloses resource sharing with re-used blocks (col. 5, line 56 to col. 6, line 4) and Killian sharing of a memory portion (col. 13, line 44 to col. 14, line 4). In view of the teachings by Ashar and the non-dependency teaching by Panchul as mentioned above, it would have been obvious for one of ordinary skill in the a14 at the time the invention was made to provide on top of such non-dependency when running in parallel and/or via speculation the step of sharing a variable in memory or a block as suggested by Panchul and enhanced by Killian to the method of optimization provided by Panchul/Killian and Ashar; because of the same reasons as to optimize resources so well known at the time the invention

was made in the art of loop pipelining and parallel execution of instructions in the art of compiler optimization techniques.

As per claim 21, in reference to claim 15, Panchul discloses using a single set of hardware resources to implement a software program entities represented by a node in a control flow, i.e. a set of hardware resources while converting a HDL function or block; into hardware implementation (e.g. Fig. 3-22).

As per claims 22 and 23, Panchul/Killian and Ashar implicitly discloses the optimization process, such as pipelining, speculation execution etc. to take place late in the phases of HDL conversion without intervention of the user (re claim 1 5).

Response to Arguments

14. Applicant's arguments filed 10/3/2005 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

Rejection under 35 USC § 102 (e):

(A) As per claims 1, 24, 34, Applicant has submitted that Panchul does not teach that a hardware configuration data directly configures a programmable logic resource (PLR) but needs a intermediate definition language – e.g. HDL - which cannot directly configure a programmable logic resource (Appl. Rmrks, pg. 14, top para, pg. 15 2nd para). The claim merely states that the hardware configuration data (HCD) directly configures a programmable logic resource; and as interpreted this HCD data does not enforce any particular type of configuration data format or language (e.g. broad term such as configuration data has no weight in terms of enforcing a particular configuration format or type per se so long as such data relate to some hardware) nor does it preclude a HDL or RTL type of constructs (language which is by itself designed to

configure/map hardware entities with defined variables representing some more concrete hardware-representing entities that are written in some programming language - see Panchul: Figs. 11-18) from reading on the HCD. It is also noted that the limitation 'programmable logic resource' is interpreted as mere software resources of the likes of variables, an address, a memory value or a node of a tree based on the analysis as set forth in the USC 112 rejection. The Applicants' argument that the HDL or RTL represents an intermediate stage as opposed to directly configuring into a PLR would also be deemed unconvincing. The rejection has pointed to the analysis using configuration data coming from HDL to map into constructs written in RTL language, all of which (e.g. *wire, pa0-16, reg*) being software resources defined for implementing hardware constructs or entities. Hence the HDL constructs being generated from the Ansi C specifications (see Fig. 2, 4) reads on compiling high level into hardware configuration data; and the deriving from HDL of RTL source code (see Fig. 5, 8) reads on HCD directly configuring some PL resource - resource interpreted herein as software constructs. The claim is not very specific about what actually constitutes the so-recited 'directly configures' limitation, e.g. in what way a configuration data configures a resource? Nor is it clear as to exactly what constitutes a programmable logic resource when the term 'resource' is being perceived as lacking a precise and/or consistent definition as per the USC 112, 1st paragraph from above. In other words, there appears to be a deficiency in the semantic aspect of the language of the claim and this is not helpful in enabling one skill in the art to possibly construe novel features of the invention. Indeed, a 'configuration data' normally serves as a means for a developer to configure a model, some design structure, or some programmatic content or skeleton thereof; but rarely is it accepted that a set of configuration data actually performs a direct configuration of

some other resource when there appears to be no particular teaching in the specifications as far as a PL resource being defined as 'directly configured' by any configuration data. There appears from reading the specifications that some configuration data is used in the operation of mapping into some hardware-representing constructs made in software; but because the claim has made it somewhat unclear (refer to Claim Objections) in conveying such teaching, hence the claim is basically not defining the claimed invention. The argument that Panchul's HDL needs to be further processed before it becomes PL resource is not deemed viable owing to all the above observations.

In the rejection, Panchul's portions are cited for showing that C language constructs are used to derive some programmatic data of the HDL format; and this HDL language being thus created is one of many interpretations that one skill in the art would adopt when construing the afore-mentioned 'hardware configuration data', on the one hand. On the other hand, since the *resource* limitation has not been defined clearly, this so-called *PL resource* is analogized to the constructs found in the RTL program whose constructs are configured from the very HDL as proffered in the cited Figures of the rejection.

(B) As per claim 20, Applicant has submitted that Examiner remarked that there no recital of a situation wherein some program is being executed to corroborate to what is recited as 'runtime instances' (Appl. Rmrks, pg. 17, bottom; pg. 18, top); and because of which the claim has been amended to provide 'hardware execution run-time' to distinguish over Killian's compilation process. But based on the analysis as set forth in the USC 112, 1st paragraph from above, this *hardware execution runtime* limitation is not disclosed sufficiently so to enable one skill in the art to construe that the inventor has possession of such at the time the invention was made. In

other words, as construed from the claim language, 'hardware execution runtime' has little patentable weight because the Examiner has to give it a broad reasonable meaning in view of the above-mentioned deficient teaching from the disclosure. The rejection points out to the fact that the meaning of 'selecting runtime instances ...' or 'execution runtime' has been viewed as a execution process for code analysis in the course of which graph analysis/generation is performed wherein patterns based on the dynamic state analysis and control flow of the hardware representation of target system that are repeated can be replaced or optimized. This runtime is no more that the dynamic analysis of the control flow of the elements (state/register therein) representing the target system hardware circuitry, these elements are software constructs defined in some TIE register/operands resource format.

Rejection under 35 USC § 103 (a):

(C) As per claims 27 and 37, Applicant has submitted that an official notice along with Examiner's citing of Panchul does not fulfill the limitation about mapping into variables as a set of wires (Appl. Rmrks, pg. 19, 2nd, and 3rd para). The claim recites 'mapping a software construct variable into a hardware construct comprising a set of wires'. Broadly interpreted, there is some software variable representing some construct such that the construct is representing for hardware component, and that such hardware representation construct 'comprises' a set of wires. Panchul has disclosed using a RTL constructs or software variables to represent what would be viewed as representing hardware constructs in the terms like *reg, wire, RAM32, pa12, module* -- see Fig. 15B1-B7; hence has disclosed the direct mapping from RTL construct into a hardware component – one of which being a set of wires -as above.

(D) Applicant has further submitted that in using Panchul, Examiner fails to provide motivation to combine (Appl. Rmrks, pg. 20, 1st para) Panchul in order teach the set of wires as propounded by the invention. The claim as it is recited does not enable a reasonable interpretation which otherwise would take into account why the use of a register as by Panchul would be detrimental as opposed to a set of wires as asserted by the above argument. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. Applicant has further argued that impermissible hindsight has been used in conjunction with the combination of Panchul with the Official Notice, i.e. such combination would not teach or suggest 'a set of wires where one of the wires indicates … value of the variable' and that there is no objective teaching from the Examiner that suggest combination based on the teaching set forth in the Official notice (Appl. Rmrks, pg. 20, 2nd para; pg. 21, 1st para). The rejection has pointed out in part of Panchul as far as the control bit to designate one wire status to be computed and that the rest of the wires to carry the combined word value of a bus signal as being taught by Panchul, and this is self-sufficient, i.e. independent of any Official Notice. That is also factual evidence not implied gathering of knowledge as alleged from above. In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347,

21 USPQ2d 1941 (Fed. Cir. 1992). In this case, Panchul's teaching of a control bit and a set of wires carrying a word composed on multi-bit has provided sufficient teaching to be fueled into the rationale as to make the above wires limitation obvious. For one skill in the art, if the reference contains sufficient teaching as to make some reasonable readjusting of the reference so that what is not explicitly disclosed therein can be modified for a good purpose without adverse effect, then the act of combining and modifying such as has been set forth in the rejection is not considered hindsight so long as the facts are found in the very prior art at hand. The argument is considered not convincing further because the Applicant has yet to convey specifics as to why the combined teachings as set forth in the rejection would not lead to expected good results.

(E) As per claims 9 and 28, Applicant has submitted that in view of Examiner's standpoint that as recited the claim does not require a *runtime execution* of any particular entity, the *runtime decisions* as claimed has been amended to further include 'at hardware execution runtime' to overcome what is perceived as 'during compilation' from Examiner's cited parts of Panchul (Appl. Rmrks, pg. 23). This added limitation has been discussed at length above for not enabling what is claimed in that it is not supported by the disclosure. The rejection has provided cited parts that fulfill both the *runtime decisions* and the *hardware execution runtime* on the basis of the alternative interpretation that has been set as a result of the rationale as set forth in USC 112, 1st para rejection. As to the argument that the Examiner only alludes to a benefit of speculation 'without providing an objective teaching ... provide motivation to modify Panchul ... decisions at runtime' (Appl. Rmrks, pg. 24, middle), the speculation teaching is factually found in Killian, and the rejection has set forth the rationale for combining based on Panchul's disclosing all the capabilities of high-level language programming to provide optimization -- such teaching being

pointed in the rejection; and this is leading to a rationale as to render obvious the speculative execution as required by the claims. Specifically, claim 9 is using the rationale from claim 6 which addresses why optimization in complex pipelining would be enhanced via speculative execution; and Applicant has yet to put forward specifics as to why the combined teachings as set forth in the rejection of claim 6 would not lead to expected good results; otherwise the argument about a prima facie case of obviousness rejection not being established would not stand. The argument that the combination from Panchul and Killian is merely hindsight using the invention as a 'blueprint' (Appl. Rmrks, pg. 25, middle) is herein referred back to section D above.

As it stands, the claims stand rejected as set forth in the Office Action.

Conclusion

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
October 27, 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100